



National Information Exchange Model

Practical Implementer's Course



United States
Department of Justice

Exchange Schemas



Practical Implementer's Course

In This Section Students Will Learn To

- Name the types of schemas used to define exchanges
- Understand the difference between document, extension, constraint, and subset schemas
- Create constraint, document, and extension schemas
- Use best practices to create different schemas



Practical Implementer's Course

Exchange Schemas

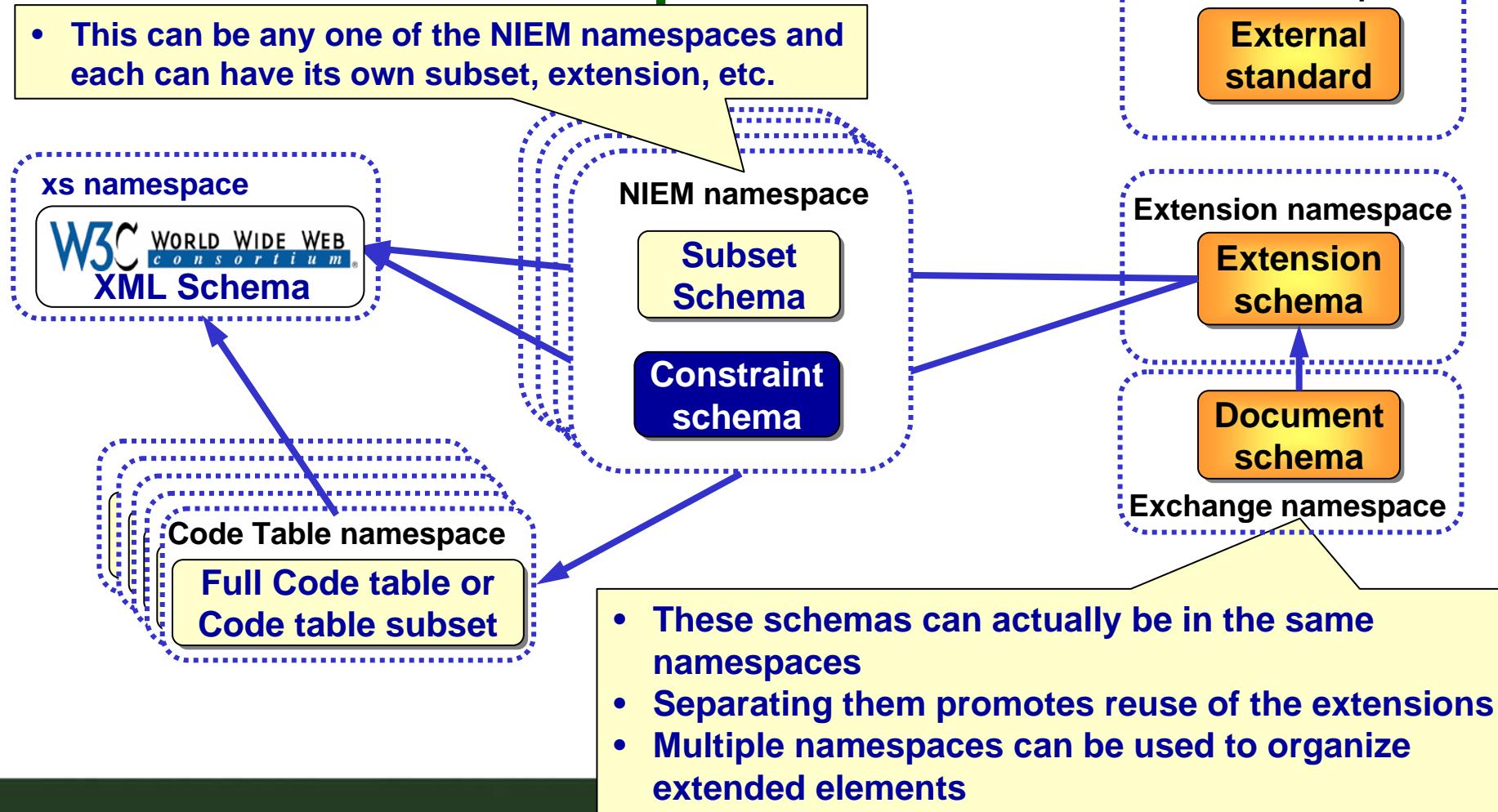
- Content is driven by output of mapping step of IEPD process
- 3 basic types
 - Extension schema
 - NIEM Data Model extensions
 - Defines some or all of data exchange structure
 - Optional, but recommended since it promotes reuse of extensions
 - Document schema
 - Defines some or all of data exchange structure
 - Constraint schema
 - Validates an exchange using basic XSD validation



Practical Implementer's Course

Schema Relationships

- This can be any one of the NIEM namespaces and each can have its own subset, extension, etc.





Practical Implementer's Course

Extension Schema

- Requires familiarity with how to extend the NIEM
- Defines some or all of data exchange structure
- Optional, but recommended since it promotes reuse of extensions



Practical Implementer's Course

Extension Schema Shell - I

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://mynamespace.com/extension/1.0"
  xmlns:ext="http://mynamespace.com/extension/1.0"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:u="http://niem.gov/niem/universal/1.0"
  xmlns:scr="http://niem.gov/niem/domains/screening/1.0"
  xmlns:j="http://niem.gov/niem/domains/justice/1.0"
  xmlns:i="http://niem.gov/niem/appinfo/1.0"
  xmlns:c="http://niem.gov/niem/common/1.0" elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xsd:annotation>
    <xsd:documentation>Sample extension schema</xsd:documentation>
    <xsd:appinfo>
      <i:ConformantIndicator>true</i:ConformantIndicator>
    </xsd:appinfo>
  </xsd:annotation>
```

Extension Namespace

Reference
required NIEM &
W3C Namespaces

Required statement of
NIEM conformance



Practical Implementer's Course

Extension Schema Shell - II

```
<xsd:import namespace="http://niem.gov/niem/universal/1.0"  
schemaLocation="niem/universal/1.0/universal.xsd"/>  
<xsd:import namespace="http://niem.gov/niem/common/1.0"  
schemaLocation="niem/common/1.0/common.xsd"/>  
<xsd:import namespace="http://niem.gov/niem/domains/screening/1.0"  
schemaLocation="niem/domains/screening/1.0/screening.xsd"/>  
<xsd:import namespace="http://niem.gov/niem/domains/justice/  
schemaLocation="niem/domains/justice/1.0/justice.xsd"/>
```

Import required
NIEM & W3C
Namespaces

<!-- New types to be added here -->

Reusable types go here

<!-- New elements to be added here-->

Reusable elements go here

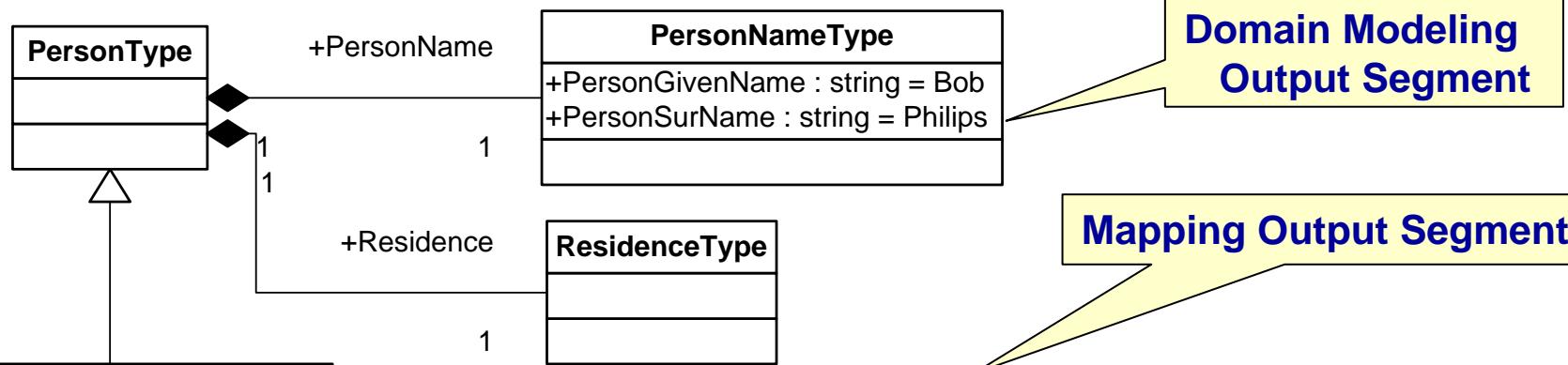
</xsd:schema>

SchemaLocation
either references:
NIEM subset or
constraint schemas



Practical Implementer's Course

Type Extension Example 1



<u>Class</u>	<u>Property or Relationship</u>	<u>NIEM Path</u>	<u>Code</u>	<u>Notes</u>
DoctorType -MedicalSchool : string = Harvard	Doctor	Doctor/MedicalSchool	new	Inherits from PersonType
	MedicalSchool	Doctor/MedicalSchool	new	
	Person	u:PersonType	none	
	PersonName	u:Person/PersonName	none	
	Residence	c:ResidentialAssociation	none	
	Residence	u:LocationType	none	



Practical Implementer's Course

Type Extension for Example 1 (continued)

- We place the following into the Extension Shell

```
<!-- New types to be added here -->
<xs:complexType name="DoctorType">
  <xs:complexContent>
    <xs:extension base="u:PersonType">
      <xs:sequence>
        <xs:element name="MedicalSchool" type="u:TextType"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent >
</xs:complexType>
```

- Not defined globally
- Cannot be reused by itself

```
<!-- New Element Definitions to be added here -->
<xs:element name="Doctor" type="ext:DoctorType"/>
```



Practical Implementer's Course

Better Type Extension for Example 1

- We place the following into the Extension Shell

```
<!-- New types to be added here -->
<xs:complexType name="DoctorType">
    <xs:complexContent>
        <xs:extension base="u:PersonType">
            <xs:sequence>
                <xs:element ref="MedicalSchool"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent >
</xs:complexType>
```

- Defined globally
- Can be reused in other structures

```
<!-- New Element Definitions to be added here -->
<xs:element name="Doctor" type="ext:DoctorType"/>
<xs:element name="MedicalSchool" type="u:TextType"/>
```



Practical Implementer's Course

XML Instance of Extension Usage

- Type Substitution

```
<u:Person xsi:type="ext:DoctorType">
  <u:PersonName>
    <u:PersonGivenName>Bob</u:PersonGivenName>
  </u:PersonName>
  <ext:MedicalSchool>Harvard</ext:MedicalSchool>
</u:Person>
```

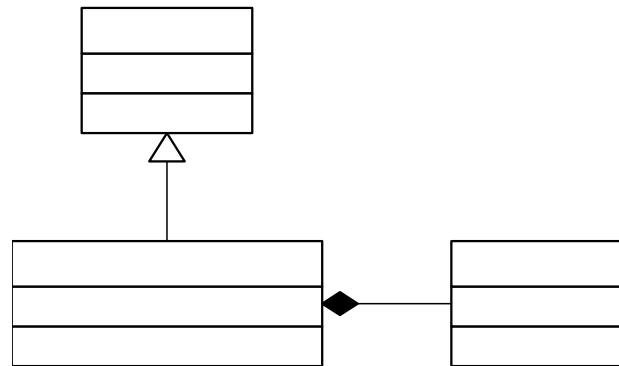
- Concrete Typing

```
<ext:Doctor>
  <u:PersonName>
    <u:PersonGivenName>Bob</u:PersonGivenName>
  </u:PersonName>
  <ext:MedicalSchool>Harvard</ext:MedicalSchool>
</ext:Doctor>
```



Practical Implementer's Course

Extension Example 2—Citation Exchange



<u>Class</u>	<u>Property or Relationship</u>	<u>NIEM Path</u>	<u>Code</u>	<u>Notes</u>
Document		c:DocumentType	none	
TrafficCitationDocument		TrafficCitationDocumentType	new	
	ExchangeSentDateTime	TrafficCitationDocumentType/ ExchangeSentDateTimeType	new	
	<i>Citation</i>	j:CitationType	none	



Practical Implementer's Course

Extension

<!-- New types to be added here -->

```
<xs:complexType name="TrafficCitationDocumentType">
  <xs:complexContent>
    <xs:extension base="c:DocumentType"/>
    <xs:sequence>
      <xs:element ref="ext:ExchangeSentDateTime"/>
      <xs:element ref="j:Citation"/>
    </xs:sequence>
  </xs:extension>
</xs:complexContent>
</xs:complexType>
```

Follows naming rules

```
<xs:simpleType name="DateTimeFormatType">
  <xs:restriction base="xs:string">
    <xs:pattern value="\d{4}-\d{2}-\d{2}(-\d{2}:\d{2})? "/>
  </xs:restriction>
</xs:simpleType>
```

Type is named and defined globally

<!-- New Element Definitions to be added here -->

```
<xs:element name="ExchangeSentDateTime" type="ext:DateTimeFormatType"/>
<xs:element name="TrafficCitationDocument" type="ext:TrafficCitationDocumentType" />
```

Element defined globally



Practical Implementer's Course

Document Schema

- Defines the root element of an exchange
- Brings together all pieces of an exchange



Practical Implementer's Course

Document Schema Shell

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://mynamespace.com/1.0/MyExchange"
    xmlns:ext="http://mynamespace.com/1.0/extension"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:import namespace=" http://mynamespace.com/1.0/extension "
        schemaLocation="ExtensionSchema.xsd" />

    <!-- New Element Definitions to be added here -->
    <xs:element name="TrafficCitationDocument" type="ext:TrafficCitationDocumentType"/>
</xs:schema>
```

Root definition for Exchange

- Exchange document defined in extension schema
- Does not necessarily have to be
- DocumentType extension could be included here instead of extension schema



Practical Implementer's Course

Constraint Schema

- Provides an additional level of constraint, beyond the subset schema
- Constraint schemas are optional
- Using them to encapsulate cardinality is usually a good idea



Practical Implementer's Course

Constraint—Encapsulating Cardinality

- Allows for cardinality and data format validation via XSD
- May simply be a copy of the subset schema with “minOccurs” and “maxOccurs” attributes set
- When validating, takes the place of the subset schema
- Cardinality can also be enforced solely within the subset schema, at the cost of easy reuse



National Information Exchange Model

Practical Implementer's Course



Reference Architecture Example: Amber Alert



Practical Implementer's Course



United States
Department of Justice

Reference resources

- On CD



Practical Implementer's Course

Summary

- Types of schemas
 - Subset, extension, document, and constraint
 - Differences between them
- Create extension, document, and constraint schemas
- Best practices
 - Define types and elements as globally accessible in extensions
 - Only define 1 root element for document schema



Practical Implementer's Course



This work is licensed under the Creative Commons Attribution-ShareAlike 2.5 License.

To view a copy of this license

- a) visit <http://creativecommons.org/licenses/by-sa/2.5/>; or,
- b) send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA."